

[illegible]

Title of the Invention: PSEUDO I/O SYSTEM AND METHOD

## PSEUDO I/O SYSTEM AND METHOD

### Background of the Invention

### Field of the Invention

5           The present invention relates to a pseudo I/O system, which is connected to a device to be tested, simulating an actual I/O system, and a method thereof.

### Description of the Related Art

10           Conventionally, there is a system comprising an I/O device connected to a SCSI or FC interface. Among others, for example, there is a system conducting various types of tests on an initiator. Its test method falls into 3 types.

15           Figs. 1A, 1B, and 1C show the conventional three test methods.

          The conventional three test methods include (1) a method conducting various types of tests by connecting to an initiator of an actual I/O device such as a hard  
20           disk drive, etc. (see Fig. 1A); (2) a method conducting various types of tests by connecting an analyzer having a test capability between an initiator and an actual I/O device (see Fig. 1B); and (3) a method conducting various types of tests by connecting to an initiator  
25           of a pseudo I/O system recited in Japanese Patent

10023379 132301

Publication No. 6-52072 or 5-73446 (see Fig. 1C).

10028279-122801

5 If various types of tests are conducted on an initiator by connecting an actual I/O device to the initiator shown in Fig. 1A with the above described method (1), all of devices to be connected such as a hard disk drive, a tape drive, etc., for which the tests are to be conducted, must be prepared. Additionally, if attempts are made to conduct a test on devices whose type is the same, for example, hard disk drives whose type is the same but whose sector sizes are different, 10 corresponding hard disk drives having different sector sizes must be prepared. Furthermore, it is extremely difficult to conduct a test on an initiator by making an abnormal operation such as no reply, timeout, etc. 15 of an actual I/O occur, even if attempts are made to conduct such a test. This is because the actual I/O does not perform an abnormal operation, when it has no problem.

20 When various types of tests are conducted on an initiator by connecting an analyzer between the initiator shown in Fig. 1B and an actual I/O device with the above described method (2), it is troublesome that corresponding actual I/O devices must be prepared if the type of an actual I/O device is changed, or if the 25 sector size of an actual I/O device is changed although

the type of the device is the same.

Although various types of tests are conducted by connecting to an initiator a pseudo I/O system, which is shown in Fig. 1C and recited in each of the above described Japanese Patent Publications, with the above described method (3), the Japanese Patent Publications target only a test on an interface such as a SCSI interface, etc. Besides, data that are handled by the pseudo I/O system and a connected test system are test data that are set in a fixed manner, and different from those for actual use. Therefore, a test for actual use cannot be conducted. Furthermore, there are problems such that (1) a system cannot run unless a test capability is added to a device to be tested; (2) if a problem that occurs in actual use depends on an address to which data is written, or contents of data, the problem cannot be reproduced, because an interface for writing test address or data is the same as that for writing an address and data when a test is conducted is the same; and (3) data other than data for actual use must be transmitted to an interface to be tested, since settings of various tests must be transmitted to a pseudo I/O system via the interface to be tested such as a SCSI interface, etc.

25

10023279-122801

### Summary of the Invention

The present invention aims at providing a system, which can be configured at low cost, simulating various types of target devices only by arranging a setting file and by changing the contents of the files, and enabling a test to be conducted while performing the same operations as those of a device in actual use by setting the setting file via an interface different from an interface to be tested.

Fig. 2 shows the configuration of the system according to the present invention, which is intended to attain the above described aim. In Fig. 2, a pseudo I/O device 1 is a device simulating an I/O device. Here, the pseudo I/O device 1 is configured by PIO (pseudo IO) processes 2, a control process 5, an adapter 7, a driver 8, etc.

Each of the PIO processes 2 controls pseudo operations such as an error of each type of an I/O device according to information set in a setting file 4. Here, each of the PIO processes is configured by a pseudo I/O unit 3, a setting file 4, etc.

The pseudo I/O unit 3 is a unit that performs processes such as a process for making an error occur according to the contents of the error set in the setting file 4. The pseudo I/O unit 3 will be described in detail

5

10

15

Next, the operations of the system are described.

20

25

3 performs a normal reply process, and returns data after being processed to the device to be tested at the request source.

At this time, the control process 5 sets as the setting file 4 the files in which the information of the I/O device to be simulated are set.

If settings are made when the setting file is referenced, after the process is performed according to the settings, the files themselves or the contents set in the files are deleted on demand, or the contents of the setting file is restored to those of normal settings of the files. As a result, a normal reply process is performed at the next time.

Additionally, if the contents of a hardware error are set in the setting file 4, the adapter 7 is made to let the hardware error occur according to an instruction from the PIO process 2. Furthermore, if the contents of an error of a protocol are set in the setting file 4, the error of a protocol is made to occur in a portion of the protocol that processes the protocol.

The timing at which a hardware or protocol error is made to occur is specified while a command is received from the device to be tested and processed.

Additionally, as the timing at which an error is made to occur, the timing at which an address to be

1.0028279 1.22801

processed and an address set in the setting file 4 match, or the timing at which an address to be processed and an error address stored when an error occurs match is used.

5           Furthermore, as the timing at which an error is made to occur, any of the moment when the contents of an error are set in the setting file 4, the timing at which data is received, the timing at which transfer data becomes a specified data transfer size while being transferred, and the timing at which a status signal is transmitted is used.

10           As a hardware or protocol error, any of a delay in a transmission start time of a frame (data), a phenomenon that part or the whole of a frame is not transmitted, a change in the contents of a frame, a change in data transfer information, a change in a data transfer method, and a change in a link state is used.

15           Accordingly, a setting file 4 is arranged in the pseudo I/O device 1 and the contents of the setting file 4 is only changed, thereby providing a low-cost pseudo I/O device, which can simulate various of types of devices to be simulated, and can conduct a test while performing the same operations as those of an actual device in use.

20  
25

10026279-122801



### Brief Description of the Drawings

Fig. 1 explains conventional techniques;

Fig. 2 shows the configuration of a system according to the present invention;

5 Fig. 3 shows the details of the configuration of the system according to the present invention;

Fig. 4 is a flowchart explaining the operations (No. 1) of the present invention;

10 Fig. 5 is a flowchart explaining the operations (No. 2) of the present invention;

Figs. 6A through 6D explain file contents and command data, which relate to the operations of the present invention shown in Fig. 5;

15 Fig. 7 is a flowchart explaining the operations (No. 3) of the present invention;

Figs. 8A through 8D explain file contents and command data, which relate to the operations of the present invention shown in Fig. 7;

20 Fig. 9 is a flowchart explaining the operations (No. 4) of the present invention;

Figs. 10A through 10D explain file contents and command data, which relate to the operations of the present invention shown in Fig. 9;

25 Fig. 11 is a flowchart explaining the operations (No. 5) of the present invention;

10028279-122801

Figs. 12A through 12D explain file contents and command data, which relate to the operations of the present invention shown in Fig. 11;

Fig. 13 is a flowchart explaining the operations  
5 (No. 6) of the present invention;

Figs. 14A through 14D explain file contents and command data, which relate to the operations of the present invention shown in Fig. 13;

Fig. 15 is a flowchart explaining the operations  
10 (No. 7) of the present invention;

Figs. 16A through 16D explain file contents and command data, which relate to the operations of the present invention shown in Fig. 15;

Fig. 17 is a flowchart explaining the operations  
15 (No. 8) of the present invention;

Figs. 18A through 18D explain file contents and command data, which relate to the operations of the present invention shown in Fig. 17;

Fig. 19 is a flowchart explaining the operations  
20 (No. 9) of the present invention;

Fig. 20 explain file contents and command data, which relate to the operations of the present invention shown in Fig. 19;

Fig. 21 is a flowchart explaining the operations  
25 (No. 10) of the present invention;

10026279-122801

Fig. 22 explains file contents and command data, which relate to the operations of the present invention shown in Fig. 21;

Fig. 23 is flowcharts explaining the operations  
5 (No. 11) of the present invention;

Fig. 24 is a flowchart explaining the operations  
(No. 12) of the present invention;

Figs. 25A through 25D explain file contents and command data, which relate to the operations of the  
10 present invention shown in Fig. 24;

Fig. 25E explains an adapter error occurrence;

Fig. 26 is a flowchart explaining the operations  
(No. 13) of the present invention;

Fig. 27 explains file contents and command data,  
15 which relate to the operations of the present invention shown in Fig. 26;

Fig. 28 is a flowchart explaining the operations  
(No. 14) of the present invention;

Fig. 29 explains file contents and command data,  
20 which relate to the operations of the present invention shown in Fig. 28;

Figs. 30A and 30B exemplify settings of an error file (process setting file) for a magnetic disk, according to the present invention; and

25 Figs. 31A and 31B exemplify settings of an error

10028279-122801

schedule file, according to the present invention.

### Description of the Preferred Embodiments

Next, preferred embodiments and their operations  
5 are sequentially described in detail with reference to  
Figs. 2 through 31.

Fig. 2 shows the configuration of a system  
according to the present invention.

In this figure, a pseudo I/O device 1, which is  
10 connected to an initiator 14 via a bus (such as a SCSI  
bus), operates as a pseudo I/O device, and conducts on  
the initiator 14 a test of determining whether or not  
a process for an error is properly performed, by making  
an error, etc. occur. The pseudo I/O device 1 is  
15 configured by a plurality of PIO processes 2, a control  
process 5, an adapter 7, a driver 8, an external storage  
device 9, a keyboard 10, a display device 11, an Ethernet  
adapter 12, an OS 13, etc.

Each of the PIO processes 2 is a process performing  
20 various types of processes as a pseudo I/O process. Here,  
each of the PIO processes is configured by a pseudo I/O  
unit 3, a setting file 4, etc.

The pseudo I/O unit 3 performs processes such as  
a process for making an error of an I/O device occur,  
25 etc. These processes will be described later with

10023379.122801

reference to Figs. 3 through 30.

The setting file 4 is a file in which information of an error made to occur in an I/O device, and the like are set. Also the setting file 4 will be described later with reference to Figs. 2 through 30.

The control process 5 performs processes such as a process for setting predetermined information in the setting file 4 of the PIO process 2, and is configured by a control unit 6, etc.

The control unit 6 receives, via a network, files where the contents of a defined error that is made to occur in an I/O device are set, and performs processes such as a process for setting the contents of the received files in the setting file 4 of the PIO process, etc.

The adapter 7 is connected to a bus using a predetermined interface (such as a SCSI interface, a fiber channel, etc.), and transmits/receives a signal.

The driver 8 transmits/receives data via the adapter 7.

The external storage device 9 stores various types of data, and holds, for example, the contents set in the setting file 4.

The keyboard 10 inputs various types of instructions, data, and characters.

10023279.122801

The display device 11 displays various items of information.

The Ethernet adapter 12 is an adapter, which is connected to a network, transmitting/receiving data.

5       The OS 13 is an operating system, which is a program controlling the whole of the system.

10       The initiator 14, to which an I/O device is connected via a bus, performs various types of processes. Here, the initiator 14 is a device to be tested, to which the pseudo I/O device 1 is connected, and on which a test of determining whether or not a process is performed at the time of an error occurrence is conducted.

15       Fig. 3 shows the details of the configuration of the system according to the present invention. This figure exemplifies the details of the configuration of the PIO process 2 and its periphery.

20       In Fig. 3, the PIO process 2 is an example of the details of the configuration of the PIO process shown in Fig. 2. The PIO process 2 is configured by an initial setting file 21, a FREE data area 22, a configuration information setting file 23, a trace file 24, an I/O data area 25, a LINK error setting file 26, a reply data area 27, an error setting file 28, a process setting file 29, an error schedule file 30, a protocol processing unit 31, a signal handler 32, an inter-process

25

communicating unit 33, etc.

The initial setting file 21 specifies a file to be used, and specifies any of the following files.

- FREE data area 22
- 5 configuration information setting file 23
- trace file 24
- I/O data area 25
- LINK error setting file 26
- reply data area 27
- 10 error setting file 28
- process setting file 29
- error schedule file 30
- protocol processing unit 31
- signal handler 32
- 15 inter-process communicating unit 33

The FREE data area 22 is a working data area.

The configuration information setting file 23 is a file where configuration information of an I/O device to be simulated, etc. are set.

- 20 The trace file 24 is a file where a history is stored.

The I/O data area 25 is an area for storing I/O data.

- 25 The LINK error setting file 26 is a file where a LINK error is set.

10068279-122801

The reply data area 27 is an area for storing reply data.

The error setting file 28 is a file where the contents of an error are set.

5        The process setting file 29 is a file where the contents of a process for an error are set.

The error schedule file 30 is a file where a schedule for making an error occur is set.

10        The protocol processing unit 31 is a unit that performs various types of processes according to the process setting file 29.

The signal handler 32 is a routine transmitting/receiving data via the driver 8.

15        The inter-process communicating unit 33 is a unit transmitting/receiving data between processes.

Next, the operations of the configuration shown in Fig. 2 are sequentially described in detail with reference to Figs. 4 through 31.

20        Fig. 4 is a flowchart explaining the operations (No. 2) of the present invention. This is a flowchart in the case where a process makes an error occur. In Fig. 4 and subsequent drawings, S1, S2, ...11, ..., S21, ..., etc. represent process orders.

25        In Fig. 4, in step S1, an operator or an external program passes, respectively via a keyboard that the

10028279.122801



operator operates or a network, files where the contents of an error are set to the PIO process. This is a process in which the control process 5 once receives the files, which are input by an operator via the keyboard 10 of the pseudo I/O device 1 shown in Fig. 2 or received from the initiator 14 via the network and the Ethernet adapter 12, and sets the received files as the setting file 4 of the PIO process 2.

In step S2, the initiator issues a SCSI command. This is a process in which the initiator 14 shown in Fig. 2 transmits a SCSI command to a bus (SCSI bus in this case).

In step S3, data is exchanged between the initiator and the pseudo I/O device based on the SCSI protocol. This is a process in which data transmission/reception (data transmission/reception such as a data write to the pseudo I/O device 1 by issuing a Write command, or a data read from the pseudo I/O device 1 by issuing a Read command) is made with the SCSI protocol (SCSI interface) under normal conditions in response to the command issued in step S2 or a command next issued.

In step S4, the PIO process executes the contents of a specified error at specified error occurrence timing. This is a process in which, for example, the

10028279.122801

PIO process 2 of the pseudo I/O device 1 executes the error contents set in the setting file 4 at the error occurrence timing specified in the setting file 4 in the above described step 1, so that the error is made to occur. As a result, the initiator 14 detects the error, and performs a process corresponding to the error (such as a process for rereading data by reissuing the Read command, for example, if the error is a read error). In this way, a test of verifying that the initiator 14 can perform a proper process for the error can be conducted on the initiator 14.

As described above, after the contents of an error are set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1 via a network or a keyboard, the initiator 14 issues a command to the pseudo I/O device 1, and performs a process corresponding to the error when the error is returned. By verifying that a process for an error is properly executed as described above, various types of tests can be easily conducted on the initiator 14. Various types of errors are sequentially described in detail below.

Fig. 5 is a flowchart explaining the operations of the present invention (No. 2). This flowchart exemplifies the operations for making an error occur by changing data. Here, a pseudo I/O on the left side

10028279 "122801

5       A pseudo I/O and an initiator in the subsequent drawings  
are similar.

In step S12, the initiator issues a SCSI command "Test Unit Ready" (see Fig. 6C).

In step S13, the PIO process 2 that has received the SCSI command changes the contents of reply data according to the error settings. With this process, the

PIO process 2 changes, for example, SCSI STATUS to BUSY (0x08), and instructs the driver to return this reply data. This is implemented in a way such that the PIO process 2 changes the contents of reply data, for example, changes the SCSI status to BUSY (0x08) according to the error contents (see Figs. 6A and 6B) set in the setting file in step S11, and instructs the driver 8 to return the changed reply data (see Fig. 6D) in response to the SCSI command received via the bus. Note that the PIO process 2 deletes the error information set in the setting file, or restores the error information to the normal state after repeating the error process once or a specified number of times in step S13. As a result, a normal reply is returned in the next process.

In step S14, the initiator that has received the reply data transmitted in step S13 performs a recovery process. This is because the received reply data is reply data where the error is set. In step S15, the initiator 14 makes a retry (reissues the SCSI command) as the recover operation of step S14, if SCSI STATUS is BUSY (0x08).

In step S16, the PIO process 2 that has received the SCSI command reissued in step S15 performs a normal operation (transmits reply data indicating normal termination).

10028279 "122801

In step S17, the initiator 14 receives the reply data indicating the normal termination, and verifies that the SCSI command is completed.

As described above, after the contents of an error are set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1 via a network or a keyboard, the initiator 14 issues a command to the pseudo I/O device 1, and makes the pseudo I/O device 1 return error reply data. Then, the initiator 14 reissues the command as the recovery process for the error. The PIO process returns normal reply data, so that the initiator 14 is normally terminated. In this way, the test program, which is not shown, within the initiator 14 can conduct on the initiator 14 a test of verifying that the initiator 104 can perform a recovery process for an error.

Figs. 6A through 6D explain file contents and command data, which relate to the above described operations shown in Fig. 5.

Fig. 6A exemplifies the process setting file 29. Namely, this figure exemplifies the contents set in the setting file 4 of the PIO process 2 in above described step S11 of Fig 5. In this example, the following items of information are set as shown in Fig. 6A.

command: Test Unit Ready

10028279.122801

valid/invalid:    valid  
 action:        control system  
 error file name: error\_file1

Fig. 6B exemplifies the contents set in the error  
 5 file having the above described name "error\_file1". In  
 this example, the following items of information are  
 set as shown in Fig. 6B.

timing:        when reply data is returned  
 error contents: changes reply data (for  
 10 example, from GOOD (0x00) to BUSY. (0x08))

The above described error contents are set in the  
 process setting file 29 and the error file 28, so that  
 the PIO process 2 can make the error occur according  
 to the error contents set in the process setting file  
 15 29 and the error file 28, namely, the PIO process 2 can  
 make the set error occur at the set error timing in  
 response to a command issued from the initiator 14. Also  
 in this case, the PIO process 2 deletes or invalidates  
 the error contents after making the error occur once  
 20 or a specified number of times. As a result, a normal  
 reply is returned at the next time.

Fig. 6C exemplifies an initiator SCSI command.  
 This exemplifies the SCSI command that the initiator  
 14 issues in the above described step S12 of Fig. 5.  
 25 The following items of information are set as shown in

100200279.122001

Fig. 6C.

command: Test Unit Ready  
contents: 00 00 00 00 00 00

Fig. 6D exemplifies STATUS. This shows the reply  
5 information when the PIO process 2 returns an error reply  
in the above described step S13 of Fig. 5. The following  
items of information are set as shown in Fig. 6D.

status: GOOD or BUSY  
value: 0 or 8

10 Fig. 7 is a flowchart explaining the operations  
(No. 3) of the present invention (No. 3). This flowchart  
exemplifies the operations for making an error occur  
by skipping the procedure stipulated in the SCSI  
protocol.

15 In Fig. 7, in step S21, an operator or an external  
test program passes, respectively via a keyboard that  
the operator operates or a network, files where an error  
occurrence due to data transfer skipping is set to the  
PIO process 2. As a result, files shown in Figs. 8A and  
20 8B to be described later are set as the setting file  
4 of the PIO process 2.

In step S22, the initiator issues a SCSI command  
"Read" (see Fig. 8C).

25 In step S23, the PIO process 2 that has received  
the SCSI command skips a data transfer, namely, the PIO

10028279 122801

process 2 does not make the data transfer, and transmits only reply data according to the error contents set in the error file. After the PIO process 2 makes such an error occur, it deletes the settings of the error file.

5 This is a process in which the PIO process 2 skips the data transfer and transmits reply data according to the error contents (see Figs. 8A and 8B) set in the setting file in step S21 in response to the SCSI command received via the bus, and deletes or invalidates the error contents.

10

In step S24, upon receipt of the reply data transmitted in step S23, the initiator detects a fault such that the data transfer is skipped.

As described above, after the error contents of data transfer skipping are set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1, the initiator 14 issues a command to the pseudo I/O device 1, and makes the pseudo I/O device 1 return error reply data. Then, the initiator 14 detects the fault such that

15

20 the data transfer is skipped, according to the error reply data. In this way, the test program, which is not shown, within the initiator 14 can conduct on the initiator 14 a test of verifying that the initiator 14 can properly detect the transfer skipping error.

25 Figs. 8A through 8D explain file contents and

10023279.1230.1



command data, which relate to the operations of the present invention shown in Fig. 7.

Fig. 8A exemplifies the process setting file 29. Namely, this figure exemplifies the contents set in the setting file 4 of the PIO process 2 in the above described step S21 of Fig. 7. The following items of information are set as shown in Fig. 8A.

```
command:    Read
valid/invalid:  valid
action:      read system
error file name: error_file2
```

Fig. 8B exemplifies the contents set in the above described error file having the name "error\_file2". In this example, the following items of information are set as shown in Fig. 8B.

```
timing:      none
error contents:  does not make data transfer
```

By setting the above described error contents in the process setting file 29 and the error setting file 28, the PIO process 2 can make an error according to the error contents set in the process setting file 29 and the error setting file 28 occur, namely an error such that a data transfer is not made, in response to the command issued from the initiator 14.

Fig. 8C exemplifies an initiator SCSI command.

This exemplifies the SCSI command that the initiator 14 issues in the above described step S22 of Fig. 7. The following items of information are set as shown in Fig. 8C.

5           command:    Read  
          contents:   28 00 01 23 45 67 00 00 01 00

Fig. 8D exemplifies STATUS. This shows the reply information when the PIO process 2 returns the error reply in the above described step S23 of Fig. 7. The following items of information are set as shown in Fig. 8D.

          status:     GOOD  
          value:      0

Fig. 9 is a flowchart explaining the operations (No. 4) of the present invention. This flowchart exemplifies the operations for making an error occur by aborting a process during the procedures stipulated in the SCSI protocol.

In this figure, in step S31, an operator or an external test program passes, respectively via a keyboard that the operator operates or a network, files where the contents of an error made to occur by aborting a transmission process are set to the PIO process. As a result, files which are shown in Figs. 10A and 10B to be described later are set as the setting file 4 of

10028279-122801



message ABORT, and performs a process for ABORT, namely, the PIO process 2 aborts the command process being executed.

As described above, after the error contents  
5 (transmission process abortion in this example) are set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1, the initiator 14 issues a command to the pseudo I/O device 1. As a result, the transmission process is aborted based on the settings of the above  
10 described error contents, so that a timeout occurs. The initiator 14 detects the timeout, and transmits the SCSI message ABORT, which is the corresponding process, to make the pseudo I/O device 1 abort the command process being executed. In this way, the test program, which  
15 is not shown, within the initiator 14 can conduct on the initiator 14 a test of verifying that the initiator 14 transmits the SCSI message ABORT to the pseudo I/O device 1 when the transmission from the I/O device is stopped, and can make the I/O device abort the command  
20 being executed.

Figs. 10A through 10D explain file contents and command data, which relate to the operations of the present invention shown in Fig. 9.

Fig. 10A exemplifies the process setting file 29.  
25 This exemplifies the contents set in the setting file

10028279 122801

4 of the PIO process 2 in the above described step S31 of Fig. 9. The following items of information are set as shown in Fig. 10A.

5           command:     Read  
           valid/invalid:   valid  
           action:       read system  
           error file name: error\_file3

10           Fig. 10B exemplifies the contents set in the above described error file having the name "error\_file3". In this example, the following items of information are set as shown in Fig. 10B.

          timing:       when data is transferred  
           error contents:   aborts data transfer after making the transfer up to 0x200 bytes.

15           The above described error contents are set in the process setting file 29 and the error file 28, so that the PIO process 2 can make the error, namely, the error such that the data transfer is aborted after being made up to 0x200 bytes, occur according to the error contents  
 20           set in the process setting file 29 and the error file 28 in response to the command issued from the initiator 14.

          Fig. 10C exemplifies an initiator SCSI command. This exemplifies the SCSI command that the initiator  
 25           14 issues in the above described step S32 of Fig. 9.

10023279 "122801

```

command:      Read
contents:     28 00 01 23 45 67 00 00 02 00

```

5            Fig. 10D exemplifies a SCSI message. This shows the information of the SCSI message that the initiator transmits in the above described step S35 of Fig. 9. The following items of information are set as shown in Fig. 10D.

Fig. 11 is a flowchart showing the operations (No. 5) of the present invention. This flowchart exemplifies the operations for making an error occur by performing an extra procedure that is not stipulated in the SCSI protocol.

In step S42, the initiator issues a SCSI command  
25 "Test Unit Ready" (see Fig. 10C).

10028279.122801

5 In step S43, the PIO process 2 that has received the SCSI command returns reply data, since this time is not error occurrence timing. This is a process in which the PIO process 2 transmits reply data according to the error contents (see Figs. 12A and 12B) set in the setting file in S41 in response to the SCSI command received via the bus, since this time is not the set error occurrence timing.

10 In step S44, the initiator receives the reply data, and recognizes that the command is completed.

In step S45, the PIO process 2 retransmits the reply data according to the error settings. This corresponds to the reply data transmission timing set in step S41 of Figs. 12A and 12B.

15 In step S46, the initiator detects the fault since it receives the unexpected reply data.

20 As described above, after the error contents such that reply data is transmitted twice are set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1, the initiator 14 issues a command to the pseudo I/O device 1. Then, the pseudo I/O device 1 transmits reply data, and the initiator 14 recognizes that the command is completed. However, since the PIO process 2 retransmits the reply data according to the setting file 4, the initiator 14 receives the second reply data,

25

and detects the fault. In this way, the test program, which is not shown, within the initiator 14 can conduct on the initiator 14 a test of verifying that the initiator 14 can detect the fault upon receipt of the  
 5 second reply data.

Figs. 12A through 12D explain file contents and command data, which relate to the operations of the present invention shown in Fig. 11.

Fig. 12A exemplifies the process setting file 29.  
 10 This exemplifies the contents set in the setting file 4 of the PIO process 2 in the above described S41 of Fig. 11. The following items of information are set as shown in Fig. 12A.

command: Test Unit Ready  
 15 valid/invalid: valid  
 action: control system  
 error file name: error\_file4

Fig. 12B exemplifies the contents set in the above described error file having the name "error\_file4". In  
 20 this example, the following items of information are set as shown in Fig. 12B.

timing: when reply data is transmitted  
 error contents: retransmits reply data

The above described error contents are set in the  
 25 process setting file 29 and the error setting file 28,

100266279-122801



5

10

```
contents:  00 00 00 00 00 00
```

15

```
value:      0
```

20

25

keyboard that the operator operates or a network, files where an error occurrence due to "endless data transfer" is set to the PIO process 2. As a result, files shown in Figs. 14A and 14B to be described later are set as the setting file 4 of the PIO process 2.

In step S52, the initiator issues a SCSI command "Read" (see Fig. 14C).

In step S53, the PIO process endlessly repeats the data transfer according to the error settings. This is a process in which the PIO process 2 endlessly repeats the data transfer according to the error contents (see Figs. 14A and 14B) set in the setting file in step S51 in response to the SCSI command received via the bus.

In step S54, the initiator detects the fault, since it receives data more than expected.

As described above, after the error contents such that data is transferred endlessly are set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1, the initiator 14 issues a command to the pseudo I/O device 1. Then, the pseudo I/O device 1 transmits data. Since the initiator 14 receives the data more than expected, it detects the fault. In this way, the test program, which is not shown, within the initiator 14 can conduct on the initiator 14 a test of verifying that the initiator 14 can detect a fault upon receipt of data

10023379.122801

more than expected.

Figs. 14A through 14C explain file contents and command data, which relate to the operations of the present invention shown in Fig. 13.

5           Fig. 14A exemplifies the process setting file 29. This exemplifies the contents set in the setting file 4 of the PIO process 2 in the above described step S51 of Fig. 13. In this example, the following items of information are set as shown in Fig. 14A.

10           command:     Read  
              valid/invalid:    valid  
              action:     read system  
              error file name: error\_file5

15           Fig. 14B exemplifies the contents set in the above described error file having the name "error\_file5". In this example, the following items of information are set as shown in Fig. 14B.

              timing:     when data is transferred  
              error contents:  endlessly transfers data

20           The above described error contents are set in the process setting file 29 and the error setting file 28, so that the PIO process 2 can make the error according to the error contents set in the process setting file 29 and the error setting file 28, namely, the error such  
25           that data is transferred endlessly, occur in response

10028279-122801

to the command issued from the initiator 14.

Fig. 14C exemplifies an initiator SCSI command. This exemplifies the SCSI command that the initiator 14 issues in the above described step S52 of Fig. 13.

5 The following items of information are set as shown in Fig. 14C.

command: Read

contents: 28 00 01 23 45 67 00 00 02 00

10 Fig. 15 is a flowchart explaining the operations (No. 7) of the present invention. This flowchart exemplifies the operations for making an error occur by making a divided data transfer.

15 In this figure, in step S61, an operator or an external test program passes, respectively via a keyboard that the operator operates or a network, files where an error occurrence due to "divided data transfer" is set to the PIO process 2. As a result, files shown in Figs. 16A and 16B to be described later are set as the setting file 4 of the PIO process 2.

20 In step S62, the initiator issues a SCSI command "Read" (see Fig. 16C).

25 In step S63, the PIO process transmits data of a divided size according to the error settings. This is a process in which the PIO process 2 transmits data of a divided size according to the error contents (see Figs.

10028279-122801

In step S64, the initiator receives the divided data.

5 In step s65, the PIO process further transmits data of a divided size according to the error settings, and continues the divided data transfer until the whole of the data is transmitted.

In step S66, the initiator continues to receive the divided data. If the initiator does not perform a process corresponding to the divided data transfer, it is determined to be an error.

10 In step S67, the PIO process transmits reply data, since it completes the divided data transfer.

15 In step S68, the initiator receives the reply data, and detects that the command is completed.

As described above, after the error contents such that divided data transfer is set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1, the initiator 14 issues a command to the pseudo I/O device 1. Then, the initiator 14 sequentially receives the divided data, and receives reply data upon termination of the divided data transfer. If the initiator does not perform a corresponding process for the divided data transfer, it is determined to be an error. If the

20 25

5

10

15

20

25

```

timing:      when data is transferred
error contents:  makes divided data transfer in
of 0x200 bytes

```

The above described error contents are set in the process setting file 29 and the error setting file 28, so that the PIO process 2 can perform a process according to the error contents set in the process setting file 29 and the error setting file 28, namely, the process for transferring divided data, in response to the command issued from the initiator 14. In this way, a test of determining whether or not the initiator 14 performs a process corresponding to the divided data transfer can be conducted on the initiator.

Fig. 16C exemplifies an initiator SCSI command. This exemplifies the SCSI command that the initiator 14 issues in the above described step S62 of Fig. 15. The following items of information are set as shown in Fig. 16C.

command: Read

contents: 28 00 01 23 45 67 00 00 0a 00

Fig. 17 is a flowchart showing the operations (No. 8) of the present invention. This flowchart exemplifies the operations for making an error occur by changing a data transfer length (size).

In this figure, in step S71, an operator or an external test program passes, respectively via a keyboard that the operator operates or a network, files where an error occurrence due to "change in data transfer

10028279 122801

length" is set to the PIO process 2. As a result, files shown in Figs. 18A and 18B to be described later are set as the setting file 4 of the PIO process 2.

5 In step S72, the initiator issues a SCSI command "Read" (see Fig. 18C).

10 In step S73, the PIO process transmits data of a size set in the error settings. This is a process in which the PIO process 2 transmits data of a set size according to the contents (see Figs. 18A and 18B) set in the setting file 4 in response to the SCSI command received via the bus.

In step S74, the initiator receives data of a size that is different from an expected size.

15 In step S75, the PIO process transmits reply data.

20 In step S76, the initiator recognizes that the command is completed, and performs a process for the case where a data size is different.

25 As described above, after the error contents such that a data transfer length is changed are set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1, the initiator 14 issues a command to the pseudo I/O device 1. The initiator 14 receives data, and further receives reply data upon termination of the data transfer. Then, the initiator detects that the data size is different, and performs a process for the case where

100636279-122801



data of a different size is received. In this way, the test program, which is not shown, within the initiator 14 can conduct on the initiator 14 a test of determining whether or not the initiator 14 can perform a process for the case where a data transfer length is changed.

Figs. 18A through 18D explain file contents and command data, which relate to the operations of the present invention shown in Fig. 17.

Fig. 18A exemplifies the process setting file 29. This exemplifies the contents set in the setting file 4 of the PIO process 2 in the above described step S71 of Fig. 17. The following items of information are set as shown in Fig. 18A.

```

command:      Read
valid/invalid:  valid
action:       read system
error file name: error_file7

```

Fig. 18B exemplifies the contents set in the above described error file having the name "error\_file7". In this example, the following items of information are set as shown in Fig. 18.

```

timing:        when data is transferred
error contents:  changes data transfer size to
0x200 bytes
The above described error contents are set in the

```

10028279.12280.1

process setting file 29 and the error setting file 28,  
 so that the PIO process 2 performs a process according  
 to the error contents set in the process setting file  
 29 and the error setting file 28, namely, the process  
 5 for changing a data length and for transferring data,  
 in response to the command issued from the initiator  
 14. In this way, a test of determining whether or not  
 the initiator 14 performs a process corresponding to  
 a change in a data transfer length can be conducted on  
 10 the initiator 14.

Fig. 18C exemplifies an initiator SCSI command.  
 This exemplifies the SCSI command that the initiator  
 14 issues in the above described step S72 of Fig. 17.  
 The following items of information are set as shown in  
 15 Fig. 18C.

command: Read

contents: 28 00 01 23 45 67 00 00 0a 00

Fig. 18D exemplifies STATUS. This exemplifies the  
 reply data that the pseudo I/O device 1 transmits in  
 20 the above described step S75 of Fig. 17. The following  
 items of information are set as shown in Fig. 18D.

status: GOOD

value: 0

Fig. 19 is a flowchart showing the operations (No.  
 25 9) of the present invention. This flowchart exemplifies

10028279.122801

the operations for making an error occur at a variation  
(No.1) of error occurrence timing in the PIO process  
2.

5 In this figure, in step S81, an operator or an  
external test program passes, respectively via a  
keyboard that the operator operates or a network, files  
where an error occurrence due to "change in reply data"  
is set to the PIO process 2. As a result, files shown  
in Figs. 20A and 20B to be described later are set as  
10 the setting file 4 of the PIO process 2.

In step S82, the initiator issues a SCSI command  
"Read (LBA=0x11111111)".

15 In step S83, the PIO process performs a normal Read  
command process (data transmission and reply data  
transmission), because the LBA (=0x22222222) set in the  
error setting file is different from that of the received  
SCSI command. This is a process in which the PIO process  
2 performs not an error process but the normal Read  
command process (data transmission and reply data  
20 transmission) in response to the SCSI command received  
via the bus, since the LBA of the received command is  
different from that in the contents (LBA=0x22222222)  
set in the setting file 4 (see Figs. 20A and 20B).

25 In step S84, the initiator recognizes that the  
process of the SCSI command Read (LBA=0x11111111) is

10028279 1.22801

completed.

In step S85, the initiator issues a SCSI command "Read" (LBA=0x22222222).

5 In step S86, the PIO process performs an operation according to the error settings, because the LBA (=0x22222222) set in the error setting file and that of the received SCSI command match. With the Read command, data transmission is made prior to reply data transmission. The data transmission is made in a normal  
10 manner.

In step S87, the initiator receives the data.

In step S88, the PIO process changes the contents of the reply data (for example, changes SCSI STATUS to BUSY (0x08)) according to the contents set in the error  
15 setting file, and transmits the reply data. The PIO process then deletes the error settings.

In step S89, the initiator receives the reply data where the error is set, and perform a recovery process for the received data where the error is set.

20 In step S90, if the received reply data includes SCSI STATUS=BUSY (0x08), the initiator makes a retry (reissues the SCSI command) as the recovery process.

In step S91, the PIO process that has received the SCSI command performs a normal operation (transmits  
25 reply data indicating normal termination).

10028279 122301

In step S92, the initiator receives the reply indicating the normal termination, and recognizes that the SCSI command is completed.

As described above, after the error contents (change in reply data) are set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1, the initiator 14 issues a command to the pseudo I/O device 1. The pseudo I/O device 1 changes reply data if the command matches the contents set in the setting file 4, and transmits the reply data. Then, the initiator 14 reissues the command as a recovery process. In this way, the test program, which is not shown, within the initiator 14 can conduct on the initiator 14 a test of determining whether or not the initiator 14 can properly perform a process for the case where reply data is changed.

Fig. 20 explains file contents and command data, which relate to the operations of the present invention shown in Fig. 19.

Fig. 20A exemplifies the process setting file 29. This exemplifies the contents set in the setting file 4 of the PIO process 2 in the above described step S81 of Fig. 19. The following items of information are set as shown in Fig. 20A.

command: Read

10028279-122801

valid/invalid: valid

action: read system

error file name: error\_file8 if LBA=0x22222222

Fig. 20B exemplifies the contents set in the above  
 5 described error file having the name "error\_file8". In  
 this example, the following items of information are  
 set as shown in Fig. 20B.

timing: when reply data is returned

error contents: change in reply data (ex:  
 10 changes from GOOD (0x00) to BUSY (0x08))

The above described error contents are set in the  
 process setting file 29 and the error setting file 28,  
 so that the PIO process 2 can perform a process (reply  
 data change) according to the error contents set in the  
 15 process setting file 29 and the error setting file 28  
 in response to the command issued from the initiator  
 14. In this way, a test of determining whether or not  
 the initiator 14 performs a process corresponding to  
 the reply data change can be conducted.

20 Fig. 20C exemplifies an initiator SCSI command.  
 This exemplifies the SCSI command that the initiator  
 14 issues in the above described step S82 of Fig. 19.  
 Items of information are set as shown in Fig. 20C.

Fig. 20D exemplifies STATUS. This exemplifies the  
 25 reply data that the pseudo I/O device 1 transmits in

10028279.122801

the above described steps S88 and S91 of Fig. 19.

Fig. 21 is a flowchart showing the operations (No. 10) of the present invention. This flowchart exemplifies the operations for making an error occur at a variation (No.2) of error occurrence timing in the PIO process.

In this figure, in step S101, an operator or an external test program passes, respectively via a keyboard that the operator operates or a network, files where an error occurrence due to "reply data change" is set to the PIO process 2. As a result, files shown in Figs. 22A and 22B to be described later are set as the setting file 4 of the PIO process 2.

In step S102, the initiator issues a SCSI command "Read (LBA=0x11111111)".

In step S103, the PIO process performs an operation according to the contents set in the setting file 4 regardless of the setting of the LBA of the firstly received SCSI command. For example, with the Read command, the PIO process makes data transmission prior to reply data transmission. This data transmission is made in a normal manner.

In step S104, the initiator receives the data.

In step S105, the PIO process changes the contents of the reply data (for example, changes SCSI STATUS to BUSY (0x08)) according to the error contents set in the

10028279-122801

setting file 4, and transmits the reply data. The PIO process leaves the set error contents unchanged, and stores the LBA (0x11111111 in this case) of the SCSI command where the error is set.

5           In step S106, the initiator receives the reply data where the error is set, and performs a recovery process for the reply data.

10           In step S107, if the received reply data includes SCSI STATUS=BUSY (0x08), the initiator makes a retry (reissues the SCSI command) as the recovery process. Namely, the initiator reissues the Read command (LBA=0x11111111).

15           In step S108, since the stored LBA and the receives LBA match, the PIO process changes the contents of the reply data (for example, changes SCSI STATUS to BUSY (0x08) according to the set error contents, and deletes the set error contents after transmitting the changed reply data. If the PIO process receives a Read command the LBA of which does not match, it performs a normal  
20 Read command process (data transmission and reply data transmission).

          In step S109, the initiator receives the reply data where the error is set after receiving normal data, and performs a recover operation for the reply data.

25           In step S110, for example, if the reply data

10028279.122801



includes SCSI STATUS=BUSY (0x08), the initiator makes a retry (reissues the SCSI command) as the recovery process. Namely, the initiator reissues the Read (LBA=0x11111111) command.

5           In step S111, the PIO process performs the normal Read command process (data transmission and reply data transmission).

          In step S112, the initiator recognizes that the SCSI command is completed.

10           As described above, after the error contents (reply data change) are set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1, the initiator 14 issues a command to the pseudo I/O device 1. With the operations of step S103 through step S112,  
15           the test program, which is not shown, within the initiator 14 can conduct on the initiator 14 a test of determining whether or not the initiator 14 can properly perform a process for the case where reply data is changed.

20           Fig. 22 explains file contents and command data, which relate to the operations of the present invention shown in Fig. 21.

          Fig. 22A exemplifies the process setting file 29. This exemplifies the contents set in the setting file  
25           4 of the PIO process 2 in the above described step S101

10028279.122801

of Fig. 21. The following items of information are set as shown in Fig. 22A.

```

command:      Read
valid/invalid:  valid
5  action:      read system
retry error:  exists
error file name: error_file9 if LBA=0x22222222

```

Unlike the case shown in Fig. 20A, the process setting file shown in Fig. 22A has the setting item of "retry error", and its content is "exists" for the above described "LBA=0x22222222". Therefore, the process transfers to the execution of the contents of the error setting file regardless of whether or not the LBA is 0x22222222.

15 Fig. 22B exemplifies the contents set in the above described error file having the name "error\_file9".

In this example, the following items of information are set as shown in Fig. 22B.

```

timing:      when reply data is returned
20  error contents:  changes  reply  data  (ex:
changes from GOOD (0x00) to BUSY (0x08))

```

The above described error contents are set in the process setting file 29 and the error setting file 28, so that the PIO process 2 performs a process (reply data change) according to the contents set in the process

25

10028279 "122801

5

10

15

15

20

25

In step S123, the initiator and the pseudo I/O

5           In step S125, the driver performs an error  
occurrence process.

As described above, after the hardware error occurrence is set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1, the initiator 14 issues a command to the pseudo I/O device 1. The PIO process 2 instructs the driver to make the error occur at the timing set in the setting file 4 while data is exchanged between the initiator 14 and the pseudo I/O device 1. The initiator 14 then recognizes the hardware error, and performs a corresponding process.

In Fig. 23B, in step S131, an operator or an external test program passes, respectively via a keyboard or a network, files where a hardware error  
25 occurrence is set to the PIO process 2. As a result,

In step S132, if the set error occurrence timing is timing that can be detected only by the driver, the PIO process passes the error settings to the driver.

In step S134, the initiator and the pseudo I/O device 1 exchange data based on the SCSI protocol.

As described above, the contents of hardware error occurrence set in the setting file 4 of the PIO process 2 are passed to the driver. The initiator 14 issues a command. Then, the driver makes the error occur upon detection of the timing set in the setting file 4 while data is exchanged between the initiator 14 and the pseudo I/O device 1. The initiator 14 recognizes the hardware error, and performs a corresponding process. In this way, the test program, which is not shown, within the initiator 14 can conduct on the initiator 14 a test of determining whether or not the initiator 14 can detect a hardware error occurrence of the pseudo I/O device 1, and can properly perform a process corresponding to the hardware error occurrence.

Fig. 24 is a flowchart explaining the operations (No. 12) of the present invention. This flowchart exemplifies the operations for making an error occur by invoking a hardware error occurrence capability prepared by the adapter.

In this figure, in step S141, an operator or an external test program passes, respectively via a keyboard or a network, files where a hardware error occurrence is set to the PIO process 2. As a result, files shown in Figs. 25A and 25B to be described later are set as the setting file 4 of the PIO process 2.

In step S142, the initiator issues a SCSI command "Test Unit Ready".

In step S143, the PIO process 2 that has received the SCSI command instructs the driver to make the error occur at error occurrence timing.

In step S144, the driver accesses an adapter register for making a hardware error occur, and makes the error occur.

In step S145, the initiator detects the error, and performs a recovery process.

As described above, after the error contents (hardware error occurrence) are set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1, the initiator 14 issues a command. The PIO process 2 makes

10023279-122801

the error occur upon detection of set timing. The initiator 14 then detects this hardware error occurrence, and performs a recovery process. In this way, the test program, which is not shown, within the initiator 14 can conduct on the initiator 14 a test of determining whether or not the initiator 14 can detect a hardware error when the error occurs in the pseudo I/O device 1, and can perform a recovery process.

Figs. 25A through 25D explain file contents and command data, which relate to the operations of the present invention shown in Fig. 24, and Fig. 25E explains an error occurrence of the adapter.

Fig. 25A exemplifies the process setting file 29. This exemplifies the contents set in the setting file 4 of the PIO process 2 in the above described step S141 of Fig. 24. Here, the following items of information are set as shown in Fig. 25A.

```

command:    Test Unit Ready
valid/invalid:  valid
action:     control system
error file name: error_file10

```

Fig. 25B exemplifies the contents set in the above described error file having the name "error\_file10". Here, the following items of information are set as shown in Fig. 25B.

timing: when reply data is returned

error contents: makes a fault occur in a signal transmitted over a cable (ex: makes Link Failure occur)

5 The above described error contents are set in the process setting file 29 and the error setting file 28, so that the error (hardware error such as Link Failure) according to the contents set in the process setting file 29 and the error setting file 28 is made to occur  
10 at error occurrence timing. In this way, a test of determining whether or not the initiator 14 can detect a hardware error, and can properly perform a recovery process can be conducted on the initiator 14.

Fig. 25C exemplifies an initiator SCSI command.  
15 This exemplifies the SCSI command that the initiator 14 issues in the above described step S142 of Fig. 24. In this example, the following items of information are set as shown in Fig. 25C.

Fig. 25D exemplifies STATUS. This exemplifies the  
20 reply data that the pseudo I/O device 1 transmits, for example, in the above described step S144 of Fig. 24. This exemplifies an error occurrence due to a register access as shown in Fig. 25E.

(e-1) in Fig. 25E shows the state where 0 (which  
25 does not make an error occur) is set in a position of

10028279 "122801



a bit that makes Link Failure occur in an adapter register. This is a state where the adapter 7 shown in Fig. 2 is made not to let the hardware error "Link Failure" occur.

5 (e-2) in Fig. 25E shows the state where 1 (which makes an error occur) is set in the position of the bit that makes Link Failure occur in the adapter register. This is a state where the adapter 7 shown in Fig. 2 is made to let the hardware error "Link Failure" occur,  
10 and the link fault occurs in the bus shown in Fig. 2. In this way, the hardware error in the above described step S144 of Fig. 24 can be made to occur.

Fig. 26 is a flowchart explaining the operations (No. 13) of the present invention. This flowchart exemplifies the operations for making an error occur  
15 due to a Lower Level protocol violation (No. 1).

In this figure, in step S151, an operator or an external test program passes, respectively via a keyboard or a network, a file where the Lower Level  
20 protocol violation is set to the PIO process 2. As a result, a file shown in Fig. 27A to be described later is set as the setting file 4 of the PIO process 2.

In step S152, if set error occurrence timing is timing that can be detected only by the driver, the PIO  
25 process passes the error settings to the driver.

10028279.122801

In step S153, the initiator issues a SCSI command "Test Unit Ready".

In step S154, the driver performs an error occurrence process at the set error occurrence timing.

5 In this example, a CLS signal is returned for a predetermined time period in response to an OPN signal.

In step S155, the initiator cannot issue the SCSI command "Test Unit Ready", since the CLS signal continues to be output in response to the OPN signal.  
10 Therefore, a timeout occurs.

As described above, after the error contents (Lower Level protocol violation) are set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1, the initiator 14 cannot issue the command (Test Unit Ready) despite attempting to issue the command. This  
15 is because the CLS signal according to the error settings continues to be output from the driver in response to the OPN signal. The initiator then recognizes that the timeout occurs. In this way, the test program, which  
20 is not shown, within the initiator 14 can conduct on the initiator 14 a test of determining whether or not the initiator 14 can perform a process corresponding to an error, namely, whether or not the initiator 14 cannot issue the command due to an occurrence of the  
25 protocol violation in the pseudo I/O device 1.

10020279.12201

Fig. 27 explains file contents and command data, which relate to the operations of the present invention shown in Fig. 26.

Fig. 27A exemplifies a LINK error setting file  
 5 26. This exemplifies the contents set in the setting file 4 of the PIO process 2 in the above described step S151 of Fig. 26. In this example, the following items of information are set as shown in Fig. 27A.

timing: when an OPN signal is received  
 10 error contents: transmits a CLS signal for one minute

Fig. 27B exemplifies an initiator SCSI command. This exemplifies the SCSI command that the initiator 14 issues in the above described step S153 of Fig. 26.  
 15 In this example, items of information shown in Fig. 27B are set.

Fig. 28 is a flowchart explaining the operations (No. 14) of the present invention. This flowchart exemplifies the operations for making an error occur  
 20 due to a Lower Level protocol violation (No. 2).

In this figure, in step S161, an operator or an external test program passes, respectively via a keyboard or a network, a file where the Lower Level protocol violation is set to the PIO process 2. As a  
 25 result, a file shown in Fig. 29A to be described later

10033279-12301

is set as the setting file 4 of the PIO process 2.

In step S162, if set error occurrence timing is timing that can be detected only by the driver, the PIO process passes the error settings to the driver.

5 In step S163, the initiator issues an ABTS frame.

In step S164, the driver performs an error occurrence process at the set error occurrence timing. Here, a BA\_RJT frame is returned in response to the ABTS frame.

10 In step S165, upon receipt of the BA\_RJT frame, the initiator detects that the instruction by the ABTS frame is unsuccessfully given.

As described above, after the error contents (Lower Level protocol violation) are set in the setting file 4 of the PIO process 2 of the pseudo I/O device 1, the initiator issues a command, and detects the error with the operations of steps S164 and S165. In this way, the test program, which is not shown, within the initiator 14 can conduct on the initiator 14 a test of  
15 determining whether or not the initiator 14 can detect an error occurrence due to a protocol violation in the pseudo I/O device 1, and can perform a process corresponding to the error occurrence.  
20

Fig. 29 explains file contents and command data,  
25 which relate to the operations of the present invention

10028279.122801

shown in Fig. 28.

Fig. 29A exemplifies the LINK error setting file 26. This exemplifies the contents set in the setting file 4 of the PIO process 2 in the above described step S161 of Fig. 28. In this example, the following items of information are set as shown in Fig. 29A.

timing: when an ABTS frame is received

error contents: transmits a BA\_RJT frame

Fig. 29B exemplifies an initiator SCSI command. This exemplifies the SCSI command that the initiator 14 issues in the above described step S163 of Fig. 28. In this example, items of information shown in Fig. 29B are set.

Figs. 30A and 30B exemplify the settings of an error file for a magnetic disk device, according to the present invention. These figures exemplify the settings for simulating similar target devices of diverse types by changing the process setting file 29. By validating or invalidating a command specific to a device, a SCSI command process of a target device can be simulated.

Fig. 30A exemplifies the settings of the process setting file for a magnetic disk device. In the file for a magnetic disk device, the following commands are set to be valid as shown in Fig. 30A.

- Test Unit Ready

10028279 "122801

- Seek
- others

Fig. 30B exemplifies the settings of the process setting file for a magnetic tape device. In the file for a magnetic tape device, the following commands are set to be valid as shown in Fig. 30B.

- Test Unit Ready
- Erase
- others

By setting a command, which is used by an I/O device to be simulated, to be valid among the commands in the process setting file 29 as described above, the process setting file 29 for the corresponding I/O device can be generated.

Figs. 31A and 31B exemplify the settings of an error schedule file according to the present invention. These figures exemplify an error schedule file 30. Here, information items such as a setting type, a command, LUN, and setting contents are set as shown in Figs. 31A and 31B.

Fig. 31A shows an example where a plurality of error schedules (schedule00) are collectively set in the error schedule file 30. Here, the PIO process collectively reads from the first setting to setting completion, and further from the setting succeeding the

setting completion to the next setting completion, so that errors are sequentially made to occur.

Fig. 41B shows an example of the error schedule file 30 (schedule01) including a series of error schedules, which is read by the PIO process.

By setting error schedules in the error schedule file 30 and by sequentially executing the schedules as described above, error processes the number of which is equal to that of set commands are automatically and sequentially executed in a similar manner as in the single error process explained with reference to Figs. 2 through 30.

With the above described pseudo I/O system and method according to the present invention, a device to be tested can be made to run by arbitrarily setting error contents that are the same as those of data in actual use, whereby a system that can be configured at low cost, and can conduct a test while making a device to be tested perform the same operations as those of an actual device in use.

In the above provided explanation of the pseudo I/O system and method according to the present invention, the test program which is not shown and makes a device to be simulated run is installed within the initiator 14, and an operator or an external test program passes,

5

10

15

20